Holistic Evaluation of Lightweight Operating Systems using the PERCU Method

William T.C. Kramer[1], Yun (Helen) He[1], Jonathan Carter[1], Joseph Glenski[2], Lynn Rippe[1], Nicholas Cardo[1]

LBNL Technical Report

Abstract: The scale of Leadership Class Systems presents unique challenges to the features and performance of operating system services. This paper reports results of comprehensive evaluations of two Light Weight Operating Systems (LWOS), Cray's Catamount Virtual Node (CVN) and Linux Environment (CLE) operating systems, on the exact same large-scale hardware. The evaluation was carried out over a 5-month period on NERSC's 19,480 core Cray XT-4, Franklin, using a comprehensive evaluation method that spans Performance, Effectiveness, Reliability, Consistency and Usability criteria for all major subsystems and features. The paper presents the results of the comparison between CVN and CLE, evaluates their relative strengths, and reports observations regarding the world's largest Cray XT-4 as well.

1. Introduction

Seldom is a leadership class system available for extended periods of time to evaluate significantly different software implementations. This paper reports the results of one of these very unique opportunities – using the 100 TF (peak) Cray XT-4 dual core system at NERSC running two completely different Light Weight Operating Systems (LWOS) – Cray's implementation of Sandia National Laboratory's Catamount Virtual Node (CVN) and Cray's Linux Environment (CLE). The NERSC XT-4 was the first platform to move fully to CLE and remains the largest platform running CLE today.

NERSC and Cray staffs were able to evaluate CVN and CLE over an extended time period, and used a comprehensive evaluation methodology called PERCU to holistically assess Hardware and Software from the perspective of the large, diverse user community that uses NERSC resources. PERCU[1], which stands for Performance, Effectiveness, Reliability, Consistency and Usability, is an evaluation methodology developed specifically for assessing systems. PERCU represents the five areas of interest to both the system managers and the user communities that make use of these systems.

This paper is organized with some introduction to the PERCU method, the Cray XT-4 hardware, and a description of the two operating systems – CVN and CLE. The introduction is followed by the results of the comparison for CVN and CLE.

1.1 Introduction to PERCU

PERCU stands for the major characteristics a user of HPC systems needs to be productive in solving science and engineering problems. The working definitions of these five categories that PERCU assesses are:

- **Performance** – factors that contribute how fast or how much work can be done on the system. Factors in this category are performance rates and amounts and/or capacities.
- **Effectiveness** – factors that relate to managing workflow on the systems so the users of the system are able to get high performance results.
- **Reliability** – factors that relate to functions, features or services that make the systems reliable and serviceable.

---

[1] The National Energy Research Scientific Computing (NERSC) Facility at Berkeley National Laboratory
[2] Cray, Inc.

- **Consistency** – factors that relate to providing consistent results, both in terms of reproducibility of answers and time to do a given amount of work.
- **Usability** –features that make systems usable to both the end users and system managers.

These five categories represent more than 84% of evaluation factors across a number of Requests for Proposals (RFPs) that have been studied, and an even larger percentage of the technical factors of those RFPs. They also include virtually all the tests – be they benchmarks or other types of tests that RFPs specify.

1.2 Introduction to the Cray XT-4

The Cray XT-4 used for this evaluation is the largest XT-4. It was acquired by NERSC as the NERSC-5 system in the first half of 2007 and named *Franklin*[ii] after America's first internationally acclaimed scientist. Franklin has 102 XT-4[iii] cabinets, with each cabinet holding 96 dual core nodes. Each core is a 2.6 GHz AMD 64-bit processor[iv] running and capable of two Floating Point Operations per clock. All nodes have 4 GB of dual channel 667 MHz DRAM running, of which 3.6-3.75 GB is user accessible. Each core has separate 64KB instruction and data L1 caches (3 cycle latency) and a 1 MB dedicated L2 cache (11 cycle latency).

Nodes are connected in a SeaStar2.2 3-D Torus of dimension 17 x 24 x 24. Each hop has a peak transport time of 53 nanoseconds. The SeaStar2 chip connects the Hyper Transport(HT) on each core to the SeaStar2 interconnect network. Each network link is capable of 7.6 GB/s peak bi-directional bandwidth while the HT is capable of 6.4 GB/s of peak bi-directional bandwidth. The SeaStar2 network is accessed through the Portals[vvi] data transport layer.

Franklin has 19,320 computational cores with 16 login nodes, 24 I/O service nodes each with two 4 Gbps Fibre Channel interfaces, four network gateway nodes each with two 10 Gbps Ethernet interfaces and 20 spare computational nodes. Approximately 450 TB of usable disk storage are directly attached to Franklin – using Lustre 1.4.6, with another 170 TB of disk mounted in a shared configuration as part of the NERSC Global File System.

1.3 Introduction to the Cray Catamount Virtual Node Operating System

On the XT-4, Cray offers two LWOSs – the CVN and the CLE for the compute nodes. CLE was also commonly known as the Compute Node Linux (CNL),. CVN[vii] is an extension of the Catamount kernel developed at Sandia National Laboratory, originally created for the single core per node Cray XT-3 systems. It uses a master-slave implementation for the dual core XT-4. CVN provides minimal functionality, being able to load an application into memory and start execution, and manage communication over the Cray Seastar Interconnect. Among many things, CVN does not support demand paging or user memory sharing, but does use the memory protection aspects of virtual memory for security and robustness, the latter to a limited extent. CVN does not support multiple processes per core, and only has one file system interface.

1.4 Introduction to the Cray Linux Environment Operation System

The CLE[viii] system, based on SUSE 9.2 during this comparison, separates, as much as practicable, computation from service. The dominant components of CLE are the compute nodes that run application processes. Service nodes provide all system services and are scaled to the level required to support computational activities with I/O or other services. The High Speed Network (HSN) provides communication for user processes and user related I/O and services.

Each CLE compute node is booted with a version of Linux and a small RAM root file system that contains the minimum set of commands, libraries and utilities to support the compute node's operating environment. A compute node's version of Linux has almost all of the services and demons found on a standard server disabled in order to reduce the interference with the application. The actual demons running vary from system to system but include init, file system client(s), and/or application support servers. CLE had specific goals to control OS jitter while

maintaining application performance. CLE uses a user space implementation of the Portals interconnect driver that is multithreaded and optimized for Linux memory management. CLE also addressed I/O reliability and metadata performance.

1.5 Evaluation Method

The evaluation period for CVN and CLE each lasted six to eight weeks between the late spring and early fall of 2007. During this time, the LWOSs were progressively presented with more challenging tests and tasks, in all the areas of PERCU. The evaluation period can be considered evolving through three phases that have different focus – albeit still approaching the system holistically. The first was a test of all functionality. Did the systems have all the features that were required and did they produce the expected (correct) results? The second phase was performance assessment when the systems were tested to determine how fast and how consistently they processed work. The third phase is an availability and performance assessment of the system's ability to run a progressively more complex workload while at the same time determining the general ability to meet the on-going system metrics. By the end of the third phase, a large part of the entire NERSC workload runs on the system, although with some limitations and a different distribution of jobs than is seen in production.

NERSC uses multiple tests to assess the performance ranging from low-level specialized subsystem component tests such as *streams*[ix] and *multipong*[x] to system wide composite tests such as the Sustained System Performance[xi] (SSP) and Effective System Performance[xii] (ESP) tests. Each test has specific goals and functions, and are selected to support each other to reduce the overall number and effort of the tests. Benchmarks are approximations of the real work a computer system can accomplish estimating the potential of the system to solve sets of problems.

Each test is made up of one computer code with one problem data set that may exhibit different behavior based on the problem being solved and the parameters involved. The tests used in this evaluation have four purposes, each one distinct. Each purpose influences the selection and the characteristics of the benchmarks as well. The four purposes of are:

1. Evaluation and/or selection of a system from among its competitors.
2. Validating the selected system actually works the way it is expected.
3. Assuring the system performance and function stays as expected throughout its lifetime (e.g. after upgrades, changes, and regular use).
4. Helping guide future system designs.

The sophistication of the approximation represented by the benchmarks depends on the fidelity needed to represent the true workload.

2. Performance –From Kernel to Sustained System Performance (SSP) Measures

Many of the subsystem component tests, kernels, and abbreviated applications used are well discussed in other reports so the reader will be directed to the references for that information. In this document, we will introduce the tests that make up the SSP since it is an important method to evaluate potency and value of the system, at different points in time.

SSP, Version 4, was used in the LWOS assessment. SSP-4 consists of the geometric mean of seven full application benchmarks: MADbench[xiii], Paratec[xiv], CAM3.0[xv], GAMESS[xvi][3], MILC[xvii],

---

[3] SSP-4 was the first time a DOE Office of Science site and the DOD sponsored HPC Modernization Program coordinated the use of the same application benchmark, GAMESS with same problem sets. This cooperation was intended to reduce the effort for bidders to provide data and to be responsive the HECRTF Workshop report, which urged government agencies to coordinate benchmark requirements. This benchmark and its data input is identical to the DOD HPCMP TI06 Gamess benchmark. NERSC and HPCMP coordinated benchmarks for the NERSC-5/TI06 RFPs.

GTC[xviii], and PMEMD[xix], each with one large problem data set. For SSP-4, the benchmarks run at differing concurrencies, ranging from 240 tasks to 2,048 tasks. The SSP-4 used more and larger application codes than any SSP to date, including one with a concurrency of 2,048. As an aside, this SSP-4 combination struck a good balance between the number and size of the benchmarks because all vendors who proposed systems for NERSC-5 provided complete data.

Application	Science Area	Basic Algorithm	Language	Library Use	SSP-4 Concurrency
CAM	Climate – Finite Volume	CFD, FFT	FORTRAN 90		56 and 240 using the finite volume method
GAMESS	Chemistry	DFT	FORTRAN 90	DDI, BLAS	64 and 384
GTC	Fusion	Particle-in-cell	FORTRAN 90	FFT(opt)	64 and 256
MADbench	Astrophysics	Out of Core Power Spectrum Estimation	C	Scalapack and large scale I/O	64, 256 and 1,024
MILC	QCD	Conjugate gradient	C	none	64, 256 and 2,048
PARATEC	Materials Nanoscience	3D FFT	FORTRAN 90	Scalapack	64 and 256
PMEMD	Life Science	Particle Mesh Ewald	FORTRAN 90	none	64 and 256

Table 2-1: Seven applications that make up the SSP-4 test.

For the purposes of this evaluation, since the underlying hardware did not change, the SSP is defined as the composite per processor performance of the each of the applications in Table 2-1 running the largest test case multiplied by the total number of cores dedicated to computational work across the entire system. The composite function used is the geometric mean as shown in Equation 2-1. Because the evaluation uses only the largest concurrency test cases, all the weights are set to 1, and number of compute cores for Franklin is 19,320, the general equation simplifies to Equation 2.2.

$$SSP = \left[\prod_{i=1}^{I} \prod_{j=1}^{J} \left(\left(\frac{\mathrm{f}_{\mathrm{i,j}}}{(\mathrm{m}_{\mathrm{i,j}} \times t_{i,j})} \right)^{w_i} \right)^{\left(\frac{1}{\sum_{i=1}^{I} \sum_{j=1}^{J_i} w_i} \right)} \right] * N$$

Equation 2-1: The equation for SSP using the geometric mean as the compositing function. $\mathrm{f}_{\mathrm{i,j}}$, $\mathrm{m}_{\mathrm{i,j}}$, and $\mathrm{t}_{\mathrm{i,j}}$ are, respectively, the number of floating point operations, the concurrency and the execution time of application i to process test case j. w_{i} is the weight assigned to application *i*, which for the assessment is set to 1 for all applications. J is 1 since the SSP uses the largest test case. N is the total number of computing cores in the system, which are 19,320 for this assessment.

$$SSP = \left[\prod_{i=1}^{7} \left(\frac{\mathrm{f}_{\mathrm{i,max}}}{(\mathrm{m}_{\mathrm{i,max}} \times t_{i,\max})} \right) \right]^{1/7} * 19320$$

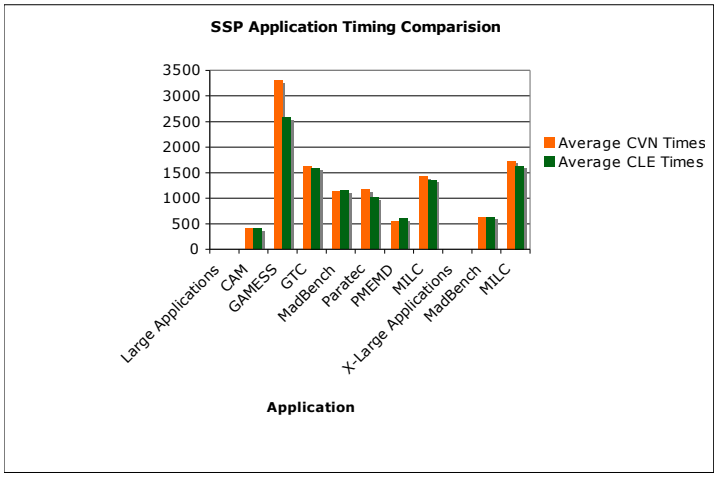
Equation 2-2: For the LWOS assessment, Equation 2.1 simplifies to this equation.

2.1 SSP Results for NERSC-5

Figure 2-1 shows the SSP-4 application run times for both CVN and CLE. The seven contributing applications to the SSP-4 metric are the five large applications (CAM, GAMESS, GTC, Paratec and PMEMD) and two X-large applications (MADBench and MILC). The run times for five of the seven SSP-4 applications are lower on CLE than on CVN. GAMESS shows the most improvement, at 22%, followed by Paratec at almost 14%. The GAMESS' CLE run time resulted from combining MPI and SHMEM communications in different sections of the code since MPI-alone or SHMEM-alone implementations ran longer on CLE than on CVN. Because GAMESS already supported MPI and SHMEM methods, it was not tremendously hard to combine the two. The need to mix communication libraries results from a different implementation of the

Portals low-level communication library on CLE and CVN that changed the relative advantages between using the MPI and SHMEM APIs. In addition, the improved Paratec timings were due in part to optimizing message aggregation in one part of the code. Two other codes, large PMEMD

Figure -2-1: The SSP-4 application run times for two LWOSs running on the same XT-4 hardware. Note that most of the runtimes for CNL are lower than for CVN.



and xlarge MADBENCH showed better run times on CVN by 10% and 1% respectively.

Figure 2-2 shows the CLE SSP performance is 5.5% better than CVN, which was surprising. CVN was in operation on multiple systems for several years before the introduction of CLE, and the expectation set by Cray and others was that using Linux as a base for a LWOS would introduce performance degradation while providing increased functionality and flexibility. The fact CLE out performs CVN, both on the majority of the codes and in the composite SSP, was a pleasant surprise and helped convince NERSC and other sites to quickly adopt CLE.

2.2 Further LWOS Comparisons

Here we digress for a moment to point out other interesting results from the comparison of CLE and CVN on the XT-4. This section briefly discusses other tests that are used at NERSC for evaluating systems.

Before proceeding, it is important to note these observations are based on data collected in the

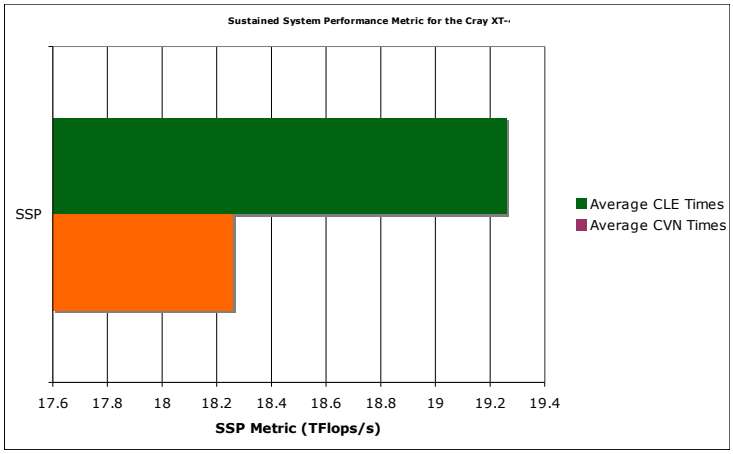


Figure 2-2: The SSP-4 metric for the same XT-4 hardware running two different LWOSs - the Catamount Virtual Node (CVN) and Cray's Linux Environment (CLE). It was a surprise that CLE outperformed CVN.

summer and fall of 2007, with the last data point being in mid October. At the time, Catamount (the single core version) and CVN had been in service for close to four years and had been through many cycles of improvements and tuning. It had also run thousands of applications at scale even though most were on single core nodes in XT-3 systems. At the time of this study, CLE was not yet released for general use, having exited development at the start of the study and was only in use on NERSC-5 for four months in by the time the last results from this study were observed. NERSC-5 was the first large scale exposure of CLE, and the study showed that there are many areas that will benefit from tuning and further improvement. However, the fact that such robust testing was feasible and the quality of the results so good for a very early operating system is very encouraging. That being said, now here are other observations about CLE and CVN.

Appendix A shows data from a wide range of performance tests for CLE and CVN. The average run times of the seven medium size application problems were slightly slower on CLE than on CVN, whereas several of the large and extra-large applications were faster. This combined with observations of early-user science applications that we do not have space to expand on, indicate codes may scale better on CLE. This was previously discussed in the Wallace paper on the design goals of CLE referenced above. Whether due to improved message handling in the node rather than the master-slave CVN is not clear.

CLE had significantly lower streams memory performance than CVN due to the Linux memory manager. This was particularly true when the test occupied only 30% of the memory. However, the streams memory rate had less impact on applications than might be expected, probably because most applications can make reasonable use of cache. Initially, several NPBs were impacted negatively on CLE, but they could be tuned using straightforward methods to match CVN performance. Full applications, as indicated above, needed little or no tuning to address memory performance differences.

I/O performance differed between CLE and CVN for the IOR benchmark and also for metadata. Lustre version 1.4.6 was used for the file system software for both CLE and CVN. For IOR, CVN did better for aggregate I/O in the initial assessment, while CLE did much better for single stream performance. The aggregate performance of CLE has since improved. Meta data performance on CLE was somewhat better than CVN.

Looking forward to the discussion of consistency in Section 5, the average Coefficients of Variation across the SSP-4 applications was 0.40% for CLE and 0.35% for CVN – remarkably close considering CLE was derived from a full blown operating system. The Coefficient of Variation (CoV) was calculated for each SSP-4 application by doing multiple runs of the same application and problem set, and then these individual CoVs were averaged. The low variability of CLE was unexpected as there was concern that increased OS jitter using Linux would decrease consistency.

Finally, the ESP-2 test, described below, ran on CLE, but never completely executed on CVN within the evaluation time period. The traditional throughput test (submitting a set of applications to over subscribe the batch job scheduler) on NERSC-5 used showed less than 1% difference between the two LWOSs.

3. Effectiveness - How Likely is Access to Performance?

Performance is only one aspect of having a system that is productive for its intended user community. The ability for users to effectively access the performance when they need it is also necessary. The ESP[xx] test was designed to provide a quantitative evaluation of parallel systems in areas not normally covered by traditional benchmarks or throughput tests but which are, nonetheless, important to production usage. There is a myriad system features and parameters that are potentially important in this regard, such as parallel launch time, job scheduling and preemptive job launch. As an alternative to assessing and ranking each feature individually, the

ESP test is a composite measure that evaluates the system via a single figure of merit, the smallest elapsed time of a representative workload that includes operational paradigm shifts.

On the Cray XT-4, ESP-2 was used to evaluate the job scheduling system for both the CVN and the CLE – both running the Torque job management system with the Moab scheduler. Multiple ESP-2 tests were performed in order to guide the adjustment of scheduling parameters. The improvements in system effectiveness rating for CLE ranged up to 22% based on improvements suggested by ESP-2.

3.1 ESP-2 Design

Overall design goals for the Effective System Performance test are:

1. Independence from the effects of CPU speed or compiler improvements on the test codes so that system management features remain the focus of the test.
2. Ability to assess the potential for a system to support different operational scheduling modes.
3. Scalability and repeatability to the test so it can be used on systems of different size and scale, as well as to compare system improvements over time.
4. Ability to reflect operational paradigm shifts.
5. Ability to reflect the performance of a scheduler as it operates with incomplete information.
6. Ability to evaluate the efficiency of job scheduling and job launching at scale.
7. Ability to encourage new features that improve a system's ability to schedule work effectively.

The ESP-2 test has been deliberately constructed to be processor-speed independent with low contention for shared resources (e.g. the file system) and be a specific measure of scalability, stability and effectiveness of a system's scheduling and resource management software. As such, it is different from a throughput test that is influenced by processors speed and compiler performance and assumes a single operational paradigm. The ESP-2 approach runs a fixed number of parallel jobs through a batch scheduler. Individually, the jobs specifically tailor their elapsed run times to closely approximate a fixed target run time. The elapsed time of the total test is independent of the processor speed and is determined, to a large degree, by the efficiency of the scheduler and the overhead of launching parallel jobs. In ESP-2, there are 230 jobs derived from a list of 14 job types, which can be adjusted if a different proportional job mix is needed. The size of each job scales with the entire system size in order to keep the test constant with regard to the number of cores. Table 3-1 shows the job types with their relative size compared to the entire system, instance count and target run time.

The ESP-2 test includes two "full configuration jobs", called Z-jobs in the test scripts, with concurrencies equal to the total number of available computational cores. The run rules for the ESP test specify the full configuration jobs cannot run at the beginning or end of the test period. The first full configuration job is only submitted after a part of the workload has already been scheduled and is running. The first Z-job has to run before any other remaining work is started. Similarly, the second full configuration job must complete within 90% of the test and not simply be the last job to be launched. The requirement to run two full configuration jobs is a difficult test for a scheduler, but it is nonetheless a common scenario in capability environments. The jobs in the ESP-2 suite are into two blocks. The first block contains all jobs except the two job type Z jobs. This first block is submitted to the queuing system in a pseudo-random order. After 40 minutes the first Z-job is submitted, and after 2 hours the second Z job is submitted. No manual intervention is permitted once the test has been initiated.

Job Type - υ	Fraction of Job Size relative to total system size - δ	Count of the number of Job Instance- κ	Target Run Time (Seconds)- π
A	0.03125	75	267
B	0.06250	9	322
C	0.50000	3	534
D	0.25000	3	616
E	0.50000	3	315
F	0.06250	9	1846
G	0.12500	6	1334
H	0.15820	6	1067
I	0.03125	24	1432
J	0.06250	24	725
K	0.09570	15	487
L	0.12500	36	366
M	0.25000	15	187
Z	1.00000	2	~100
Total		230	

Table 3-1: The ESP-2 Job Mix. The amount of work ESP-2 performs is based on system scale.

The fractional-size is the size of the job as a fraction of total system size. For example, if the system under test has 1024 cores for computation, then the size of job-type B is 64 (= 0.06250 x 1024) cores. The ESP-2 test can be applied to any system size and has been verified on 64, 512, 2048, 6726 and 19,320 computational core systems.

For the purposes of this discussion, it is useful to define the ESP unit of computational "work" as the product of the run time of a job and job size (number of cores). Following our example, job-type B is designated 64 CPU x 322 seconds = 20,608 CPU seconds of work. For a system with 1,024 processors, and not counting the Z type jobs since their time will slightly vary based on system size, the work is 11,031,792 CPU seconds.

Given a total amount of work, ω_s, a hypothetical absolute minimum time, with N being the total number of computational processors in the system (T-BEST) can be computed by dividing the work by the system size. In the example above, T-BEST = 10,773 seconds (~ 3 hours). T-BEST is independent of the total system size and the processing speed of the system. The ESP efficiency ratio ε is defined as the T-BEST divided by the observed elapsed time of the ESP-2 test. This is the metric of the ESP test. For increasingly efficient systems, the ratio approaches unity.

The T-BEST is simply a convenient definition of a lower bound. It is not possible to obtain T-BEST in a real test even in the optimal case. Therefore, most attainable ESP-2 ratios fall in the range of 0.6 - 0.8.

Equation 3-1: The Effectiveness ratio is the time the test actually runs compared to the time

$$T-BEST = \omega = \sum_{n=1}^{\upsilon} N * (\delta_n * \kappa_n * \pi_n)$$

$$\varepsilon = \frac{T-BEST}{\sum_{i=1}^{I} (\chi_i * T_i)}$$

the best packing solution indicates.

The ESP-2 and instructions for installation are located at http://www.nersc.gov/projects/esp.php.

3.2 ESP-2 and the NERSC-5 Cray XT-4

Once the job scheduler, launcher and resource manager were functional with CLE, ESP-2 was used to tune and improve the software components. Figure 3-1 shows a chart of one of the earlier ESP-2 runs, which took 14,882 seconds. The target time on *Franklin* was 13,671 seconds - reflecting about a 75% rating. In these charts (created by Sarah Anderson of Cray Inc.) the job colors are matched to the job sequence number in the ESP-2 test.

The tick marks on the X-axis are in intervals of 1,000 seconds. The target test time of 13,671 seconds is indicated by the third dashed line. The first and second dashed lines are Z job

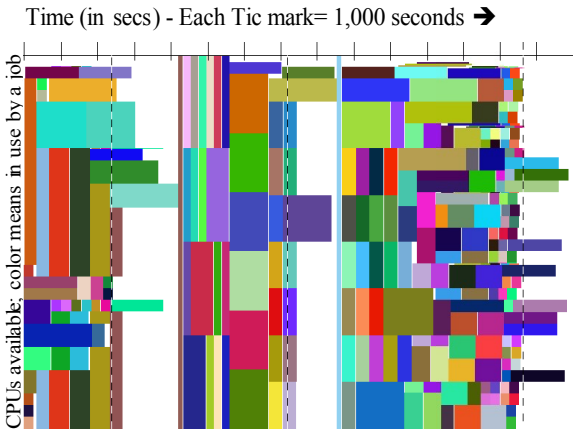


Figure 3-1: An ESP-2 run on NERSC's Cray XT-4. The Y axis is the number of CPUs used in the systems (19,320 compute processors). The X-axis is time from the start of the test – 0 at the left. Hence a job with a concurrency of 1,024 takes and 1,000 second long is represented by a colored rectangle 1,024 points high and 1,000 seconds long.

submission times. The test run in Figure 3-1 shows a large amount of white – indicating long periods where many processors were idle, lowering effectiveness. Observe the system was starting many large scale (many tasks) jobs early in the test, and deferring the longer running jobs.

Compare Figure 3-1 with another test represented by Figure 3-2. The test runs in 12,156 seconds – 22% faster. The difference is a better selection of the longer running jobs earlier. The Figure 3-2 test was done after changing two MOAB parameters. RESWEIGHT was changed from 0 to 1 and WALLTIMEWEIGHT was set to be 1. These changes allowed jobs to be launched in a more deterministic order than the previous selections with the longest job first, if other job characteristics are equal. The result is the duration of the drain periods drops significantly and that there are many fewer times with idle CPUs.

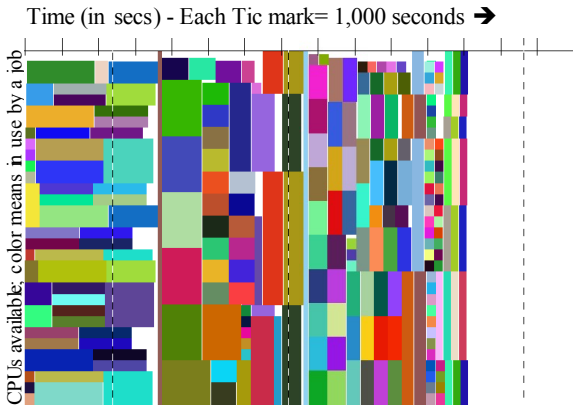


Figure 3-2: Another ESP-2 run with priority placed on longer running and larger jobs. This test confirms the system can be effectively scheduled and was significantly faster than the target time.

The scheduling priority used in Figure 3-2 is aligned with NERSC operational scheduling, which favors the highest concurrency jobs above all others. This policy is needed to overcome the default scheduling of the smaller jobs because it is easier for a scheduler to accumulate resources for smaller jobs, and reflects NERSC's role as a capability high-performance computing center. The ESP-2 test showed the XT-4 with Torque/Moab/ALPS was capable of running the NERSC workload effectively.

3.3 Additional ESP-2 Results for NERSC-5

In addition to the discussion about setting scheduler parameters to meet the expected time, it is noteworthy that ESP-2 made other contributions. ESP-2 ran on both CVN and CLE versions of the system software.

For the CVN runs, ESP-2 encountered a minor obstacle since it uses standard Linux/Unix system calls within for each task to get the time. Each task then uses that time returned to calculate how long it should run since jobs are self-terminating. CVN provided no mechanism for a task to get system time or time of day, so a new routine was added to the ESP-2 jobs. This was just a minor inconvenience. More importantly, ESP-2 failed on CVN a number of times. These failures were due mostly to the large number of nodes in use during the test. A variety of hardware and software problems were detected using ESP-2 as a blunt diagnostic.

ESP-2 on CLE also brought to light a number of problems, particularly with the early versions of Torque, which had just been ported to the CLE environment. CLE used an entirely new resource manager – the Application Level Placement Scheduler, ALPS – which replaced the resource manager on the CVN systems. The interaction between Torque and ALPS needed to be refined and ESP-2 helped identify the length of time it takes to start jobs, the load balancing for the job scheduling nodes and other issues. Furthermore, the ESP-2 workload continued to uncover infant mortality of hardware components.

Thus, ESP-2 was an excellent stress test in its own right, in addition to validating the job scheduling and launch softwares' effectiveness.

4. Reliability

The reliability and availability characteristics differed between CLE and CVN. CVN was more sensitive to individual hardware component failures within the system, including single node failures. The number of system-wide failures was higher for CVN. CLE enabled more components to have redundant features. For example, the Lustre file system, version 1.4.6, was part of both operating environments. However, if Lustre detects a failure of hardware components or OSTs, it can use alternate paths but since CVN is a polling LWOS, Lustre cannot to indicate the change in roles to the computational nodes. CLE provides the ability to make use of redundant paths. Other failures in CVN also generated system wide impact, while under the CLE environment, these turned into failures of job.

Memory errors under CLE became more obvious. CLE uses standard Linux memory management on the compute nodes, where CVN uses a simplified memory manager. Since hardware vetting was going on during the initial CVN testing, we expected early hardware failures to be mostly node failures. Despite having the hardware error rate stabilize, it was surprising to see a large number of memory errors occur when the CLE evaluation began. This was suspected to be due to the difference in how CLE lays out memory. The number of applications during this time that used more than 80% of the full memory was less than 20%. CVN memory allocation is linear and very repeatable, but CLE was more likely to try to use memory areas that had not been previously been exercised. Over time, with some hardware adjustments as well as more CLE use, the memory errors under CLE stabilized to the same rate as CVN.

4.1 Comparison of CLE with Other Full OS Systems at NERSC

Failure data for all NERSC systems from 2001 to 2006 was assembled from the NERSC operational trouble ticket system where operations and systems staff record all system outages and issues[xxi]. In addition to the operational logs, data was accumulated from paper records of repairs

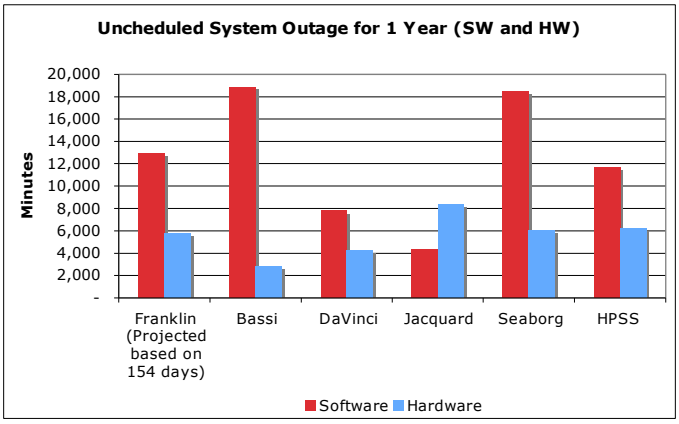


Figure 4-1: Total number of unscheduled downtime for the major NERSC systems over a 1 year period. All systems other than Franklin are from 2006. Franklin is a partial year - 154 days spanning late 2007 and 2008 which is projected to a full year for comparison.

kept by operations staff, vendor repair logs and automatic operating system error logs. The data was assembled for analysis in a *mysql* database. Each data record was manually reviewed and correlated with other information so the database is as consistent as possible. Redundant and overlapping records were combined. Furthermore, each event was reviewed to determine the most likely subsystem category that generated the error.

The NERSC failure data is available at a web site – http://pdsi.nersc.gov - as part of the Petascale Data Storage Institute SciDAC research collaboration. The web site allows interactive queries, charting and exporting of the data to CSV formatted files.

The NERSC systems covered during this time period were the IBM SP 3+ *Seaborg* (2001), the IBM SP 5- *Bassi* (December 2005), the Linux Networx AMD/IB cluster *Jacquard* (July 2005), the SGI Altix 3200 *Davinci* (September 2005), the High Performance Storage System *HPSS* (2003), the NERSC Global Filesystem, *NGF* (October 2005) and the commodity cluster system *PDSF* (2001). The dates show the beginning of the data collection period in the data base, which corresponds to the date of production or 2001 (the start the data collection).

The *Franklin* Cray XT-4 failure data, which begins in October 2007 is not part of the database yet, but has been compiled separately and correlated with the data in the database. This enables comparison of the two largest NERSC systems – Seaborg with 6,756 cores and Franklin with 19,576 cores.

4.1.1 Software and Hardware Errors

Analysis of the data shows for five of the six systems, software is the primary cause of down time of the entire system. In some cases the amount of time a system is down due to software is more than five times that of hardware generated outages. Figure 4-1 shows this data as the percent of unscheduled downtime for six major NERSC systems. Franklin, Seaborg, Bassi, Jacquard Davinci and PDSF are computational systems of various architectures and HPSS is a large data archive. Franklin had been in service for less than ½ a year at the time of this analysis – 154 days to be exact -- from October 26, 2007 to March 28, 2008. For comparison, the Franklin times are projected to a full year by multiplying by 2.37.

There are several aspects that should be considered in the comparison. First, *Franklin* and *Bassi* were in their initial period of operation, while all the other systems were in operation for at least a year before the data collection period. Hence, it may be expected that the number of outages for *Franklin* and *Bassi* reduce for later periods. This is exactly what happened to Bassi. Comparing the downtime between 2006 and 2007 indicates the hardware downtime decreasing by more than a factor of 2 and the software improving by more than 6 times. However, even with those improvements, software still caused 2.4 times more downtime than hardware. Second, only Jacquard shows more hardware downtime than software. This is in part due to a continuing problem with memory components during 2006, which produced significant downtime.

The assignment of an outage to the hardware or software category – and in the next section to the subsystems – is not foolproof. Root cause analysis was not done for all failures but instead the most likely cause was assigned. For example, it may be that some software outages had an underlying hardware cause that contributed to the failure but was not reported. The assignment of the failure category is guided by type of corrective action taken by the system managers and the vendor support personnel.

4.1.2 Subcomponent Error Analysis

Looking more closely at the two largest systems available, *Seaborg* and *Franklin*, it is useful to compare outage times by subsystem for system wide failures. A system wide failure is one where the entire system is unable to meet its service commitments. Individual failures must not degrade system performance sufficiently to cause a system wide failure. NERSC uses the following definition for system wide failure.

An entire system is considered down if the system is unable to process work at a specified level. Many components in the system have redundancy including spare compute nodes and login nodes, alternative routing in the interconnect and for I/O access. A system wide outage occurs if any of the following requirements cannot be met:

- Complete a POSIX ‘stat’ operation on every file within all file systems and access all data blocks associated with these files.
- Complete a successful interactive login on at least 75% of the login nodes in the system. Failures in the LAN do not constitute a system-wide failure.
- Run the NERSC benchmark suite for that system, including the full configuration test.
- Sufficient file system bandwidth is available and all files are accessible.
- Full Interconnect bandwidth is available. For systems that can route messages in multiple paths, some links or paths may be out of service but only to a negotiated limit.
- All nodes have access to external networks and bandwidth is at least 75% of total network I/O node bandwidth.
- User applications can be submitted, launched and/or completed via the job scheduler.
- Other failures that reasonably disrupt work on a large portion of the nodes.
- A spare node is unavailable when a compute node fails. The number of spare nodes is negotiated based on the total number of compute nodes.

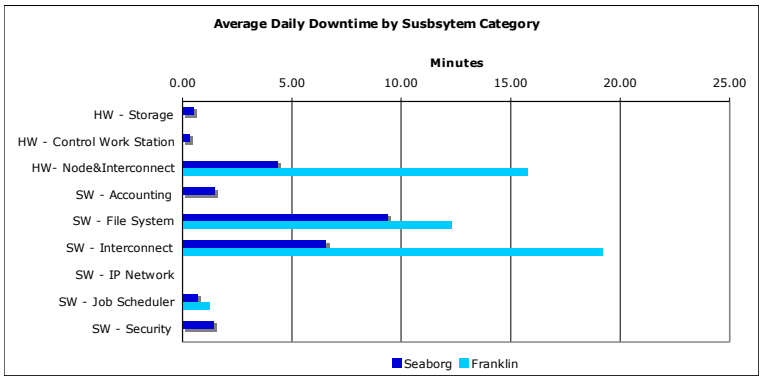


Figure 4-2: Average Daily Downtime by Subsystem Category for both systems.

The Hardware category is separated in three major areas – Node and Interconnect hardware, Storage hardware, and Control hardware. Software failures are placed into seven categories – Accounting, Filesystem, Interconnect, IP Networking (externally), Job Scheduling, Security, and Various. The *various* category covers things like license servers and mis-configurations. Figure 4-2 shows outage and recovery time for Seaborg and Franklin. Because Seaborg was in service more than 10 times longer than Franklin, the absolute times would be on different scales, so outage time is normalized to daily average outage. Figure 4-2 shows the two systems together in a normalized comparison of downtime, the amount of downtime was divided by the total time period of the data collection giving the average minutes of downtime per day. Seaborg has outages in more categories that may be due to the longer time it was in service. Seaborg also has less average downtime than Franklin because the number of outages per unit time was less, not because the length of any given outage was less.

Comparing the charts indicates the majority of hardware problems are node and interconnect and not the shared storage. Seaborg has local disks, whereas Franklin does not. Both systems suffer the majority of their software outages from interconnect software and file systems. A word of caution is that symptoms of other subsystem failures, such as interconnects and nodes, can exhibit as file system failures since the file system is spanning all components.

Figures 4-3 shows the Mean Time To Repair by subsystem for both systems. The data is independent of the data collection period, and the scale is the same. MTTR is calculated as the total downtime divided by the number of incidents. Overall the outages on Franklin are significantly shorter because it is possible to bring Franklin up in less than 25% of the time it took to boot Seaborg. The large MTTR for Seaborg Security is the result of two events, one of which required a complete system build that took more than a week because of the complexity of Seaborg's rebuild process.

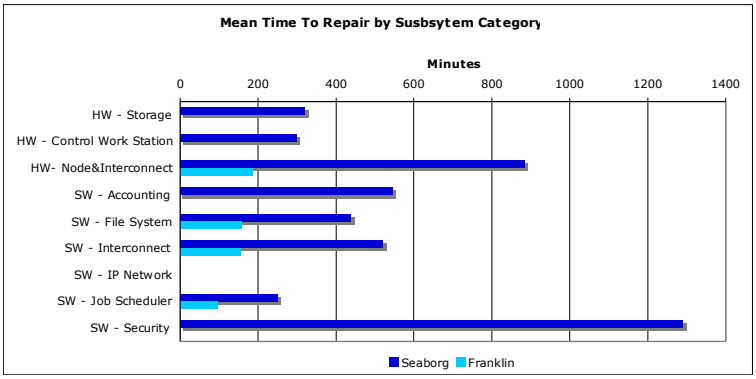


Figure 4-3: The Mean Time to Repair by Subsystem Category for two very large systems.

4.2 Job Completion

As noted above, some system wide outages under CVN became job failures under CNL. Job success rates were collected and analyzed. The basic logistics of evaluating job completion was more complicated than first envisioned. Initially, CLE provided inconsistent and incomplete error messages due in part to incorrect error message propagation across layers of the software stack. Many inconsistencies were resolved within the evaluation period so more accurate error messages were being produced. The completion status of jobs is traced from logs and system process logs. Table 4-1 shows the job success rate between Sept 18, 2007 and April 11, 2008 – more than ½ a year from the early acceptance testing of CLE through production usage. Unlike other job completion metrics that assess how often the exact same job completes successfully, this metric

deals with all jobs running on the production system. During this time, 178,133 significant computational jobs ran on the system, Jobs that failed due to a system wide error outage or were killed before a scheduled outage are excluded.

Failure Error Category	**Number of Jobs**	**Percent of Jobs**
SUCCESS - Job clearly succeeds	117,884	66.2%
WALLTIME - Job ran to the wall clock time limit. A number of users let the job run out of time intentionally. However, there are cases where a node assigned to the job is in ill health but has not yet been detected, causing the job to go very slowly or hang with no progress.	12,614	7.1%
WIDTH - A mismatch between what the job requested and what the aprun command uses -– normally a user error	0	0.0%
NODEFAIL – A node assigned to the job failed or crashed – possibly hardware.	192	0.1%
UNEX - This error indicates MPI buffers need to be increased.	75	>0.05%
ENOENT – A requested executable file does not exist.	1,148	0.6%
LIBSMA - An error within the SHMEM communication library	70	>0.05%
SIGTERM - Job received a Terminate Signal (Kill -9) This could have been from the user or the system	58	>0.05%
NOAPRUN - The batch job did not appear to execute an aprun. This is usually due to a batch scripting error.	6,516	3.7%
NOTRACE – For some unidentified reason, process accounting data could not be traced to identify the aprun associated with this job. The job did execute an aprun but the parent process id was 1 so it could not be properly matched. The usually cause is that a job was killed and the last process to exit was aprun so its ppid was 1	11,389	6.4%
QUOTA - Job exceeded a File System quota	2,865	1.6%
ATOMIC – The job failed due to a software problem when using parts of the SHMEM library (the problem has since been fixed)	4	>0.05%
UNKNOWN - The status of the job completion was non-determinate. What is known is the aprun command had a non-zero exit code. This may be due to a system problem but more likely due to some user action that prevents recording the exit status in system logs, e.g. an application trapping a signal or redirecting I/O.	25,318	14.2%
Total	178,133	

Table 4-1: Job Failure Error Categories and Data.

A subset of the failing jobs – several hundred – was manually analyzed in detail. Users who submitted the job were contacted to determine if the failure was intentional, in the application, or system generated. This investigation led to several conclusions:

1. Root cause job failure analysis is very time consuming for support staff and users, so it is not tractable to do a full analysis of every job failure. Automation is required.
2. NERSC has sophisticated users who can determine the cause of errors in their runs and proactively report suspect job failures that are not due to their error.
3. Many errors were due to user mistakes or code problems. However, each category had job failures due to system issues assigning a single category to only user problems is not possible.

a. For example, many WALLTIME errors were under user control, but occasionally a hung node or other undiagnosed error caused jobs to start, make no progress for their entire time slot, and exit, giving the same error.
b. Over running file quota typically is considered a user error, but the system generated 49 such errors since January 2008 despite having the quota function entirely turned off while awaiting bug fixes. This is a system-generated error.

4. WALLTIME, WIDTH, SIGTERM, NOAPRUN, are now considered likely user generated unless there is a pattern detected when many user jobs generate the same error. QUOTA will be in the category once it is functional.
5. NODEFAIL and ATOMIC is clearly a system issue.
6. UNKNOWN and NOTRACE represent a significant number of failures and is troubling since it means an exit status could not be automatically determined. It should be possible for the system to reliably record all process exit codes for post mortem analysis. It remains a goal to drastically reduce the number of unknown conditions for job exits.

Job completion metrics were unexpectedly difficult to accurately assess in the automated manner that is necessary on such a large system. Work continues to more accurately report and diagnose errors. Despite the difficulties, tracking job exit codes is valuable. At the moment, we are looking for patterns such as large increases in the percent of a particular category, which then merits then manual investigation.

5. Consistency of Performance

A useful metric for understanding consistency is the Coefficient of Variation (CoV), defined by the standard deviation of a sample divided by the arithmetic mean. The CoV has shown to be very useful in a number of situations in diagnosing consistency issues on real systems[xxii, xxiii, xxivxxv]. Specifically, for a given number, O, of application performance results that show performance of t-obs$_o$ on a given system, the Coefficient of Variation is defined as:

$$\overline{t-obs} = \frac{1}{O}\sum_{o=1}^{O}\left(t-obs_o\right)$$

$$CoV = \frac{\sqrt{\frac{1}{O}\sum_{o=1}^{O}\left(t-obs_o-\overline{t-obs}\right)^2}}{\overline{t-obs}}$$

Equation 5-1: The Coefficient of Variation is the standard deviation divided by the mean of a series of observations.

The average CoV across the SSP-4 applications was 0.4% for CLE and 0.35% for CVN – remarkably close considering CLE was derived from a full-blown operating system. The CoV was calculated for each SSP-4 application by doing multiple runs of the same application and problem set, and then these individual CoV's were averaged.

However, other tests show significant decreases in consistency with CLE, particularly shorter running tests such as the NAS Parallel Benchmarks. Streams, particularly the version of the streams test that use less than 50% of available memory, showed increased variability as well as lower performance. If the ratio of CLE CoV to CVN CoV for all tests – from single core to the full configuration test - are averaged, CLE has a CoV six times that of CVN. This is opposed to about a 14% increase for the larger scale SSP applications. It is important to note the CLE CoV is still more than a factor of five lower than that observed on other systems that run full blown Linux or Unix based operating systems on different hardware.

Both CLE and CVN provide very consistent timing for applications. Under CLE, the consistency actually seems to improve with scale. Both CVN and CLE provide an improvement in consistency over systems that use full operating systems on compute nodes.

6. Usability

Both CVN and CLE used in this study had full-featured programming environments[xxvi], including PGI, Pathscale, and Gnu compilers for Fortran, C and C++ codes; Cray's Portals communication layer that supports MPI and SHMEM parallel programming models; a rich set of Cray LibSci and AMD Core Math libraries; Cray performance and profiling tools; modules environment for managing system and custom built software; Torque/Moab for batch system managements; and Lustre parallel file system.

Usability was assessed for both ease of use by the computational community and ease of management for system managers. 377 separate criteria were examined for CLE and CVN Expected features were tested for functionality as well as performance. Of the 377 items, 254 were testable for this analysis – with 35 applying only to future functions and 88 being more descriptive and not testable. More than half, 53%, of these criteria related to Usability, with 39% focused on user Usability issues and 14% on System Manager Usability. Table 6-1 shows the comparison of how many usability features were operational between CVN and CLE. Less than 10% of the items under CLE have issues, almost all regarding modest to slight discrepancies with performance. Only one of the 134 Usability tests is currently outstanding for CLE - proper functioning of disk quotas which is current a high priority problem report.

	CVN	CLE
Number of features tested	248	254
Number of features properly working	232-90.5%	232 – 91.3%

Table 6-1: Initial Usability Tests for CVN and CLE.

Further, usability was assessed by moving large scale applications to the systems – work done in conjunction with early users. The usability advantages of CLE over CVN are a bigger set of standard POSIX C library routines for compute node applications, so users have more control for their applications, and less need to rewrite the source codes. CLE's increased OS functionality simplifies code porting from other platforms than CVN. At least in some cases, compilations are quicker. CLE provides other needed functions, such as OpenMP, pthreads, Lustre failover, and the possibility of adding Checkpoint/Restart and other features. CLE enabled more options for debugging tools, such as the Allinea DDT (Distributed Debugging Tool), which is now the operational debugger running on Franklin.

Some disadvantages of CLE over CVN are the increased memory footprint for OS so that it leaves less usable memory space for user applications. The difference is about 170 MB/node from our measurements (about 4.25% of the available memory). MPI latency for farthest intra-node is higher under CLE (8.12 μsec) than CVN (7.55 μsec), although this may improve for future CLE OS releases.

NERSC launched an early user program on Franklin during the CVN and CNL assessment period. We worked with experienced users on Franklin to benefit all parties. Many early users were able to run high concurrency jobs to tackle much larger problem sizes and model resolutions that were impossible before. Users got a chance to get hands-on a new architecture and a relatively lightly loaded system, and user jobs were free of charge from their allocations. Running the broader range of user applications helped find any problems (and fixes) in the system. The overall user feedback for CLE was very positive, even at its early exposure. Most applications were relatively easy to port to Franklin, the user environment (via modules) was familiar, and the batch system worked well.

7. Conclusions and Observations

The PERCU holistic approach to system evaluation proved valuable in differentiating the Performance, Effectiveness, Reliability, Consistency and Usability characteristics of two different operating systems running on the same hardware. Both CVN and CLE proved functional Light Weight Operating Systems when tested in this comprehensive manner.

CLE showed benefits over CVN in performance, scalability, reliability and usability, while showing only slight, acceptable decreases in consistency. The full benefits of one or the other could not be exposed with evaluation of only one dimension such as performance. The ability to test different system software at scale on the same hardware is extremely beneficial to all parties.

The use of PERCU for evaluation speeded the introduction of CLE at scale and assured the Franklin system would be able to serve the needs of a diverse and large science community.

8. Acknowledgements and Thanks

This work was supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.. We also wish to thank our co-workers at NERSC and Cray for all the hard work deploying Franklin and solving many problems during the time periods of these tests.

Appendix A. CVN and CLE Times and Rates for all Performance Tests

			Average CVN Results	CVN CoV	Average CLE Results	CLE CoV	Ratio CLE/CVN
Kernels							
Serial NPBs	v2.3 Class B		NPB not the final				
	BT	Mflops	612.78	0.21%	627.66	0.36%	102.4%
	CG	Mflops	259.95	0.05%	347.74	0.09%	133.8%
	FT	Mflops	520.45	0.30%	648.12	1.02%	124.5%
	LU	Mflops	658.67	1.81%	716.12	0.30%	108.7%
	MG	Mflops	720.67	0.18%	717.05	0.72%	99.5%
	SP	Mflops	380.6	0.31%	390.46	1.49%	102.6%
64 way NPBs	V 2.4 Class D						
	BT	Mflops	576.12	0.02%	584.44	0.42%	101.4%
	CG	Mflops	81.97	0.25%	106.07	0.07%	129.4%
	FT	Mflops	348.02	0.31%	552.83	0.30%	158.9%
	LU	Mflops	679.42	7.00%	716.49	0.53%	105.5%
	MG	Mflops	785.02	0.05%	958.63	0.07%	122.1%
	SP	Mflops	350.67	0.02%	403.52	0.08%	115.1%
256 way NPBs	v 2.4 Class D						
	BT	Mflops	585.54	0.04%	575.9	0.13%	98.4%
	CG	Mflops	112.08	0.17%	165.77	1.80%	147.9%
	FT	Mflops	302.1	0.79%	467.96	0.35%	154.9%
	LU	Mflops	1107.05	0.03%	1037.93	0.59%	93.8%
	MG	Mflops	667.72	0.25%	802.95	0.15%	120.3%
	SP	Mflops	340.32	0.05%	387.58	0.41%	113.9%
Streams - 30%		MB/sec	7326.43	0.01%	6470	0.90%	88.3%

Memory/ 1 Core							
Streams - 60% Memory/1 Core		MB/sec	7130.23	0.01%	6469.8		90.7%
Streams - 60% Memory/2 Core		MB/sec	3606.2	0.01%	3445.7		95.5%
Multipong-Min		Microseconds	4.65	0.92%	5.94	2.49%	127.7%
Multipong-Max		Microseconds	7.55	0.30%	7.75	1.76%	102.6%
Multipong DC Farthest Node		Microseconds			8.11	0.45%	
Multipong DC Intranode		Microseconds	2.08	0.41%	2.72	0.41%	130.8%
Multipong DC Nearest Node		Microseconds			6.15	0.38%	
Meta Data - Aggregate - Dedicated		IOPs	6658.04	1.83%	7199	18.32%	108.1%
Meta Data - Single - Dedicated		IOPs	4518.08	2.03%	4280	1.35%	94.7%
Meta Data - Aggregate - Multi-use		IOPs					
Meta Data - Single - Multi use		IOPs					
IOR - I/O Aggregate - Read		MB/s	12644.01	1.02%	11411.7	4.85%	90.3%
IOR - I/O Aggregate - Write		MB/s	9634.7	0.30%	4197.44	1.65%	43.6%
IOR - I/O Single - Read		MB/s	592.93	1.65%	1299.97	1.48%	219.2%
IOR - I/O Single - Write		MB/s	535.91	21.87%	926.26	32.88%	172.8%
Full Config		Seconds	24.63	0.38%	24.86	0.44%	100.9%
Medium Applications							
CAM		Seconds	1591.77	0.02%	1605.84	0.07%	100.9%
GAMESS		Seconds	2683.67	0.02%	3269.23	0.13%	121.8%
GTC		Seconds	1504.54	0.01%	1469.72	0.32%	97.7%
MadBench		Seconds	1281.78	0.05%	1269.73	0.15%	99.1%
Paratec		Seconds	692.09	0.05%	699.95	0.57%	101.1%
PMEMD		Seconds	450.5	0.13%	457.33	0.46%	101.5%
MILC		Seconds	191.86	0.16%	194.02	0.17%	101.1%
Large Applications							
CAM		Seconds	407.09	0.05%	405.54	0.17%	99.6%
GAMESS		Seconds	3297	0.16%	2572.18	0.47%	78.0%
GTC		Seconds	1636.47	0.06%	1590.01	0.33%	97.2%
MadBench		Seconds	1132.4	0.14%	1153.73	0.20%	101.9%
Paratec		Seconds	1171.67	0.11%	1017.06	0.26%	86.8%
PMEMD		Seconds	543.25	0.18%	598	0.88%	110.1%
MILC		Seconds	1421.48	0.94%	1355.35	0.05%	95.3%

X-Large Applications							
MadBench		Seconds	627.5	0.19%	635.35	0.17%	101.3%
MILC		Seconds	1735.65	2.05%	1629.85	0.20%	93.9%
SSP		Tflops/s	18.26		19.26	0.21%	105.5%
Throughput		Seconds	1979.5	3.32%	1993	2.00%	100.7%
ESP		Seconds			13497	1.40%	
Consistency		%					
OS Jitter - EP -					0.263		
OS Jitter - FT -					0.574		
Average SSP CoV				0.40%		0.35%	

References

[i] A full explanation of the PERCU method is available in the draft PhD Dissertation – www.nersc.gov/~kramer/PhD/draft_dissertation

[ii] http://www.nersc.gov/nusers/systems/franklin/

[iii] *Cray XT Series System Overview,*, Cray, Cray inc., S-2423-20 , the *Cray XT-4 Data Sheet* - http://www.cray.com/downloads/Cray_XT4_Datasheet.pdf and the Cray XT Overview - http://www.cray.com/products/xt4/index.html

[iv] LBL-62500. John Levesque, Jeff Larkin, Martyn Foster, Joe Glenski, Garry Geissler, Stephen Whalen (Cray, Inc); Brian Waldecker (AMD); Jonathan Carter, David Skinner, Helen He, Harvey Wasserman, John Shalf, Hongzhang Shan, and Erich Strohmaier (2007). Understanding and Mitigating Multicore Performance Issues on the AMD Opteron Architecture.

[v] Brightwell, Ron, Rolf Riesen, William Lawry, Arthur Maccabe, *Portals 3.0: Protocol Blocks for Low Overhead Communication,* Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS '02) http://www-static.cc.gatech.edu/classes/AY2008/cs8803hpc_spring/papers/brightwell-portals.pdf

[vi] Brightwell, Ron, Rold Riesen, Authur Maccabe, *Desing, Implemtation and Performnce of MPI on Portals 3.0,* http://www.cs.unm.edu/~maccabe/papers/p3-mpi-journal.pdf

[vii] Wallace, David, *Compute Node Linux, New Frontier in Compute Node Operating Systems*, 2007 Cray User Group Conference, May 7-10, 2007 Seattle, Washington

[viii] Wallace, David, *Cray XT-3/XT-4 Software Status*, 2007 Cray User Group Conference, May 7-10, 2007 Seattle, Washington

[ix] http://www.cs.virginia.edu/stream/

[x] Reference for Multipong

[xi] Kramer, William T.C., PERCU Results in a Reawakended Relationship for NERSC and Cray, Cray User Group (CUG 2007) Conference, Seattle, WA, May 2007.

[xii] Wong, Adrian T., Leonid Oliker, William T. C. Kramer, Teresa L. Kaltz, and David H. Bailey, System Utilization Benchmark on the Cray T3E and IBM SP, 5th Workshop on Job Scheduling Strategies for Parallel Processing, May 2000, Cancun, Mexico.

[xiii] Oliker, Lenoid, J. Borrill, J. Carter, D. Skinner, R. Biswas, Integrated Performance Monitoring of a Cosmology Application on Leading HEC Platforms, International Conference on Parallel Processing: ICPP 2005.

[xiv] http://www.nersc.gov/projects/paratec/

[xv] http://www.ccsm.ucar.edu/models/atm-cam/index.html

[xvi] M.W.Schmidt, K.K.Baldridge, J.A.Boatz, S.T.Elbert, M.S.Gordon, J.H.Jensen, S.Koseki, N.Matsunaga, K.A.Nguyen, S.J.Su, T.L.Windus, M.Dupuis, J.A.Montgomery J. Comput. Chem. 14, 1347-1363 (1993).

[xvii] http://www.physics.indiana.edu/~sg/milc.html

[xviii] W. W. Lee, *Gyrokinetic particle simulation model*, Journal of Computational Physics, v.72 n.1, p.243-269, September 1, 1987 [doi>10.1016/0021-9991(87)90080-5]

[xix] http://amber.scripps.edu/pmemd-get.html

[xx] Wong, Adrian T., Leonid Oliker, William T. C. Kramer, Teresa L. Kaltz, and David H. Bailey, *Evaluating System Effectiveness in High Performance Computing Systems*,. Proceedings of SC2000, November 2000 Postscript | PDF

[xxi] Schroeder, Bianca, Garth A. Gibson *Understanding Failures in Petascale Computers*.. SciDAC 2007. Journal of Physics: Conference Series 78 (2007) 012022. - Also see http://www.pdsi-scidac.org/

[xxii] Kramer, William and Clint Ryan, *Performance Variability on Highly Parallel Architectures*, the International Conference on Computational Science 2003, Melbourne Australia and St. Petersburg Russia, June 2-4, 2003

[xxiii] Skinner, David and William Kramer, *Understanding the Causes of Performance Variability in HPC Workloads*, 2005 IEEE International Symposium on Workload Characterization (IISWC-2005), October 6-8, 2005, Austin, Texas.

[xxiv] Kramer, William and Clint Ryan, *Performance Variability on Highly Parallel Architectures*, the International Conference on Computational Science 2003, Melbourne Australia and St. Petersburg Russia, June 2-4, 2003

[xxv] Skinner, David and William Kramer, *Understanding the Causes of Performance Variability in HPC Workloads*, 2005 IEEE International Symposium on Workload Characterization (IISWC-2005), October 6-8, 2005, Austin, Texas.

[xxvi] DeRose, Luiz and John Levesque, *Tools and Techniques for Application Performance Tuning on the Cray XT4 System*, a tutorial at the 2007 Cray User Group Conference, May 7-10, 2007 Seattle, Washington